Termination Project, Professor Harold Lewis

Dynamic Granularity in Word Sense Induction via Transformers and Growing Neural Gas

Grady Kurpasi

[grady.kurpasi@outlook.com](mailto:grady.kurpasi@outlook.com) (213) 304-8541

|  |  |  |
| --- | --- | --- |
| A R T I C L E I N F O  19 Nov 2021  Keywords: Word Sense Induction, Python, Pytorch  Transformers, BERT  Growing Neural Gas  Azure, Cloud,  Word2Vec, Semeval 2015 |  | A B S T R A C T  This paper documents an initial foray into machine learning and computational linguistics research. The research goal of the paper was to explore the application of Growing Neural Gas as a clustering strategy in Word Sense Induction to achieve dynamic granularity in sense identification. The engineering goal of this paper was to build an extensible and scalable environment for further research in computational linguistics. Productive results of the application of Growing Neural Gas to Word Sense Induction were not achieved however, a reusable environment for continuing research was constructed and is ready for continuing exploration of this topic.  Code at: [GradyKurpasi/word\_sense\_induction (github.com)](https://github.com/GradyKurpasi/word_sense_induction) |

1. Introduction

This paper documents research conducted at State University of New York, Watson College of Engineering under the Systems Science department. The efforts herein are a formal first step in exploring computational linguistics via current state of the art technology. The aims of this work were:

1. Explore the application of the Growing Neural Gas (GNG) algorithm to sense clustering in Word Sense Induction (WSI) to achieve dynamic sense granularity.
2. Develop an extensible and scalable software environment suitable for continuing research in computational linguistics.

The work did not provide definitive positive or negative indications of the applicability of Growing Neural Gas to Word Sense Induction. The work did however provide a research environment suitable for follow-on research in the topic as well as continuing research in computational linguistics. The organization of the remainder of this paper is as follows:

* 1. Introduction & Organization

Describes the content of the remaining sections of this paper and appendices.

* 1. Background

Describes the motivation behind this work. Long term motivation revolves around improved automatic semantic analysis of text. Immediate motivation is improving the granularity and flexibility of word sense disambiguation (i.e., figuring out what a word means in context).

* 1. Experimentation

Describes the approach taken to engage this topic. Discusses why Growing Neural Gas was chosen as a focus of research and provides an overview of the technologies chosen to implement the research environment.

* 1. Results and Analysis

Describes experimental results and analyzes why initial experiments were non-productive. Compares the clustering strategy used herein to the clustering strategies employed in current state of the art work and traditional methods.

* 1. Next Steps

Describes a roadmap for follow-on work in Word Sense Induction and computational linguistics in general

* 1. Appendices

The appendices provide a detailed description of the technologies and methods employed in this work and a blueprint for recreating the research environment.

1. Background
   1. The Big Picture

We live squarely in an age of ‘Big Data’. The massive body of digital artifacts we generate provide insight into the complex, often invisible systems that drive our species. Big Data allows individuals to collect, visualize and analyze information on the scale demanded by the modern world. It provides us with both introspection and foresight that would otherwise by beyond the scope of single or even multitude human beings. The means to do this though remains in the realm of maths. We continue to crack the code of high speed, massive scale processing in the generally unambiguous language of numbers.

What continues to elude us is full scale automatic processing in the languages of humans. Human language, both spoken and written, is notoriously ambiguous and often inextricably intertwined with other modes of symbolic communication. As such, the grand database of human written and spoken language remains resistant to large scale semantic interrogation and an age of ‘Big Knowledge’ remains tantalizingly out of reach.

* 1. Word Sense Disambiguation

There are many impediments to machine processing of human language. The most fundamental perhaps is Word Sense Disambiguation (WSD). Simply, Word Sense Disambiguation is determining what a word means in context. Human languages have words that are homophones: words that are individually indistinguishable in spoken form (e.g., wright and right), words that are homographs: words that are individually indistinguishable in written form (e.g. bass: fish and bass: instrument), and words that are a mix and match of both (e.g. board: noun, board: verb, bored: adj). In practice, humans disambiguate words via context, generally constituted by proximal linguistic content, but also possibly extending into extra-linguistic or even environmental cues. In this work, only textual content is considered.

Word Sense Disambiguation is deceptively complex. Just precisely defining what actually constitutes a ‘*word*’ remains debated and theoretical. A precise definition of ‘*meaning*’ is equally enigmatic. This paper considers a word to be a string of alphabetic characters bounded by space characters on both sides. In English, this is a fair distinction, and this is also the generally accepted unit of analysis in existing WSD work. Likewise, meaning is considered to be a mapping of a text-string word to a previously defined sense (definition) that signifies the intended communicative content of that text-string. More simply, word sense disambiguation maps a word to one of its definitions.

* 1. Previous Approaches

J. R. Firth stated in 1957 that “you shall know a word by the company it keeps” (Firth 1957). This beautiful description of context states that the communicative content of a text symbol like ‘board’ can be determined by the linguistic environment it occurs in, that is, by the words that surround it and its relationship to them.

The idea that context determines, or at least constrains, meaning underlies most computational approaches to WSD. Early statistical models of language employed N-Grams. N-Grams modeled the probability of another word or meaning appearing within n words of a target word. Thereby, predictions could be made that if ‘board’ occurred within n words of ‘Directors’ it referred to a collection of people rather than a plank of wood. This line of reasoning continues to underly WSD efforts today.

Statistical language models dominated WSD efforts for some time but did not produce results powerful enough to drive general semantic computing. Eventually, statistical models gave way to machine learning approaches which greatly improved WSD effectiveness, but which likewise fell short of empowering general semantic computing.

* 1. The Problem With WSD

Early efforts in applying machine learning to WSD revealed two limitations. The first was a limitation in the algorithms being employed. Initial efforts conducted ‘supervised’ learning, meaning that they required an answer key in the form of hand-tagged corpora to train. Semantically tagged corpora are resource intensive to produce. Many of those initially employed were decades old by the time machine learning was applied to them. This limitation was addressed by developing methods to apply unsupervised learning to WSD. Unsupervised learning requires no answer key, just vast amounts of text. This enabled analysis of modern electronic stores of text.

The second limitation was tied to human word sense disambiguation itself. As ML-WSD matured, it was pointed out that measuring progress was difficult because different research efforts employed different sense inventories making it very hard to compare the accuracy of results (Raganato, Camacho-Collados, & Navigli, 2017). It was also observed that even when working off the same sense inventory, human annotators frequently disagreed with each other. Some analyses show Inter Annotator Agreement as low as 67-80% (Navigli, 2009).

This is a result of the nature of language and how we perceive it. Different individuals perceive varying degrees of nuance and granularity in word senses. In fact, there is little consensus on the active inventory of words and senses in a language.

The Merriam-Webster online English dictionary provides 20 glosses (senses) for ‘board’. The Cambridge Dictionary online however provides 35 with significant overlap. Dictionary.com provides 26 with significant overlap, and the Urban Dictionary provides 7 with no overlap. There is effectively no way to develop a definitive sense inventory due to these nuances.

Additionally, language is a living system. Social, cultural, generational and domain specific employment of words results in endless novelty and nuance that can reflect everything from subject expertise to social in-group/out-group status.

* 1. Word Sense Induction

Given the limitations of initial WSD strategies, researchers developed other methods of determining what words mean. Word Sense Induction is currently the leading alternative approach.

Word Sense Induction still relies on the concept that a word’s meaning is illuminated by the linguistic context it appears in. It differs from standard WSD in that it employs unsupervised learning to search for context patterns.

A WSI algorithm will analyze the linguistic context of every occurrence of a target word (e.g., co-occurrence with other words, part of speech, etc.). It will then ‘cluster’ (or group) these contexts based on similarity to each other. Each context cluster is considered to represent a distinct sense of the target word. These senses can then be mapped via supervised learning to existing sense inventories.

WSI’s ability to operate in an unsupervised manner improves on WSD by not requiring large hand tagged corpora. Some implementations can operate without hand tagged corpora at all. In general, WSI is better suited to large bodies of modern text (e.g., internet corpora).

It appears current state of the art in WSI/WSD was achieved by Bevilacqua and Navigli in 2020 (Bevilacqua, Navigli 2020). Their approach combines machine learning with a knowledge-based approach. Knowledge base approaches use rules and discrete word knowledge to inform sense selection. Unsurprisingly, Navigli and Bevilacqua are two leading minds behind BabelNet a multilingual lexicographic and encyclopedic language resource.

For simplicity, this paper does not address Bevilacqua & Navigli’s latest approach. Instead, this work examines the previous state of the art achieved by Amrani and Goldberg (Amrani, Goldberg 2019). Amrani and Goldberg employ a purely machine learning approach that capitalizes on the success of Transformers in language processing (described later).

Amrani and Goldberg’s Language-model Substitutions with Dynamic Patterns (LSDP) approach differs from previously described methods. It relies on a trained language resource to suggest substitutions for a target word. These substitutions are still generated based on context, but it is the sets of substitutions associated with each target word occurrence that are clustered to identify senses. The method of clustering is the topic of examination in this paper.

* 1. Dynamic Clustering

When the authors first introduced LSDP (Amrani, Goldberg 2018) they employed a clustering strategy that produced a fixed number of clusters. The authors identified that this was suboptimal and recommend a method for producing a dynamic number of clusters in their follow-on work (Amrani, Goldberg 2019).

For every target word, they set a limit of 10 senses and then induce this number of clusters. Every occurrence of the target word is given fuzzy membership in all 10 senses. The sense that is most probable for an occurrence is said to dominate that occurrence. Any sense that dominates 2 or more occurrences of the target word is considered strong. Any sense that only dominates 1 occurrence of a target word is considered weak and is discarded. That occurrence is then assigned membership in the next nearest cluster / sense. This removes noise (i.e., over-specification) from the LSDP model and allows the number of word senses recognized to arise from the data. The authors were not the first to recommend dynamic clusters but were the most successful in implementing it. Many WSI approaches still employ fixed clustering.

* 1. Research Motivation: Dynamic Sense Granularity

The research motivation behind this paper is (ironically) not to improve WSI/WSD accuracy, but to improve flexibility in the granularity of senses produced by WSI/WSD. As previously discussed, different individuals perceive varying degrees of nuance in word senses. For example, for most people ‘*board*’ might refer simply to a *skateboard* or *Board of Directors*. For specialists, ‘*board*’ might refer to up to 18 different types of skateboards (Scooterlay 2021) or up to 7 different types of directorates (Medium 2019).

The ‘correct’ degree of nuance desired will likely depend on the downstream application of WSI/WSD. In disambiguating text and determining reference, especially in domain specific applications, high degrees of nuance are likely desired within the domain area, and lower degrees desired outside of the domain area. The remainder of this paper addresses clustering strategy with the aim of achieving dynamically scalable sense granularity.

1. Experimentation
   1. Goal

Develop a means to achieve dynamically scalable sense granularity in text disambiguation

* 1. Methodology

The methodological reasoning behind this project was inspired by Word2Vec and WordNet. It also draws on the success of Transformers in language processing. Engineering-wise it capitalizes on several successful technologies including cloud computing and graph databases.

This project aimed to find an encoding for word contexts similar in spirit to Word2Vec. The aim was to encode the context of word occurrences (i.e., the linguistic environment a target word appears in) in an n-dimensional hyperspace such that similar contexts would be spatially proximate. This coordinate scheme would be used to support sense clustering as well as to provide a persistent means of addressing both contexts and senses.

The project also aimed to model hyponymy/hypernymy and troponymy relationships between senses/contexts in the same manner as WordNet to play a part in organizing sense granularity.

The general idea was that modifying the clustering strategy in WSI could provide senses with increasing degrees granularity (i.e., specificity) and that persisting sense details at every level of granularity to a database would provide downstream applications with a means of dynamically scaling the specificity of WSD.

* 1. Growing Neural Gas

Growing Neural Gas was introduced by Bernd Fritzke in 1995. GNG was selected as a clustering algorithm because it requires no parameters to determine the number of clusters developed; all clusters arise from the data. GNG is particularly well suited to mapping topologies in high dimensional data, and it can conduct continuous learning on streaming data (Fritzke 1995). Additionally, it appeared that GNG could efficiently cluster all words in a text at the same time.

GNG creates a network of nodes and edges that map to input space. The algorithm incrementally moves the network toward non-empty areas of the input space. The algorithm inserts nodes and edges in areas of the network where error is high (i.e., adaptation to input signals is low) and also prunes edges and nodes that lie in empty input space.

The methodological intent behind the application of Growing Neural Gas was to investigate both nodes and clusters at each timestep with the expectation that cluster centroids would gravitate toward sense centers and that nodes or groups of nodes would eventually approximate finer grained sense centers. The goal was to examine the distance in input-space from cluster centroids and node locations to both input signals (the text and contexts being disambiguated) and solution keys (the hand tagged senses provided with the corpora)

The assumption was that as the GNG network passed a certain threshold from random initialization to input signal adaption that the location in input-space of nodes and cluster centroids would encode semantic information and that as the GNG network progressed toward finer and finer adaption that the location of nodes and cluster centroids (presumably increasing in number to fit the data) would encode more granular semantic information.

At each time step the intent was to record the vector representations of nodes and cluster centroids and assign all input signals fuzzy membership in clusters and node association (based on a input-space distance parameter).

The reasoning was that the progression of input word’s cluster membership and node proximity would inform establishment of hyponym and troponym relationships (which generally indicate that X is a more specific instance of Y) and that this could be extended as well to establishing relationships between senses in the answer key

The final reasoning for selection of GNG is that it can conduct continuous learning. That is, it will continue to adapt to modifications in input signals as they are introduced. Clustering on new data will not require re-running the algorithm from the beginning. This was attractive in that it enables the network to detect changes in word usage over time.

* 1. Python

This project was primarily implemented in Python. Python was chosen for the speed of development it permits, the robust number of libraries frameworks available and the wide community and industry support it enjoys.

Python was unveiled by Guido van Rossum in 1991 (Van Rossum 2021). Python is a high level, scripted programming language that features dynamic typing and automatic garbage collection. These features make prototyping code in python fast. Additionally, it is easy to find and install libraries that extend the capabilities of the base language.

The tools used to develop Python for this project included Visual Studio Code, Anaconda and GitHub. Microsoft’s Visual Studio Code is a lightweight code editor that features a plug-in environment. VS Code plug-ins provide add-on features and facilitate integration with several other technologies including GitHub, Docker and Microsoft Azure.

Anaconda is distributed by Anaconda Inc. Anaconda packages Python releases with a large number of python libraries that generally facilitate data science research and development. Additionally, Anaconda includes Conda a tool for managing virtual environments. Python virtual environments allow python scripts to utilize specific versions of the language and libraries. Managing virtual environments is crucial to code reuse and portability. Virtual environments also make it easier to migrate development between machines.

GitHub is an online code repository. GitHub’s primary use is code versioning and backup, but GitHub also has extensive code sharing and collaboration features. These features combined with Python’s virtual environments make it easy to explore and test other developers’ code. A VS Code plug-in integrates GitHub functionality directly into the code editor.

* 1. Transformers

Transformers are a key technology in this project. Transformers are a neural network architecture that have been proven effective in language processing. Transformers were first introduced by Vaswani et al in 2017. They employ a strategy of self-attention that, in language processing, relates every word in a sequence to every other word in a sequence. Vaswani et al’s transformer uses multiple self-attention layers allowing words to ‘attend’ to multiple aspects of relation to other words (e.g., reference, modification, syntactic function, etc.). Finally, Vaswani et al’s transformer employed an encoder / decoder architecture that was immediately applicable to language tasks like automatic translation.

Transformers quickly replaced recurrent neural networks as the go-to architecture in language processing because they achieved state of the art results in most language tasks and also increased the amount of parallelization possible in processing. This made them faster and computationally cheaper.

Transformers also conduct unsupervised learning; this and their lower computational cost spurred a new paradigm in language processing. Previously endpoint applications would adopt a neural network architecture and train it directly on domain specific text. Transformers spurred use of pre-trained models with application-specific fine-tuning ‘heads’ attached to process domain specific text. These became popular due first to their effectiveness but also due to the prohibitive overhead of training on the vast amounts of text most commercial transformers are trained on.

The first widely popular pre-trained transformer was Open AI’s Generative Pre-Trained transformer, now called GPT-1 (Radford et al 2018). This was followed quickly by Google’s Bi-directional Encoder Representations from Transformers (BERT) (Devlin et al 2018). Subsequent iterations of BERT and GPT remain prevalent in language processing. Amrani and Goldberg employed BERT in their 2019 release of their LDSP model. Bevilacqua and Navigli also make use of BERT in their 2020 work.

This work likewise used the BERT (small uncased) transformer to generate context embeddings. Three strategies were used to do this. In the first two strategies single sentences were used for sequence encodings. BERT processes sequences of equal length so these sequences had to be padded to facilitate processing. In the first encoding, sentences were left aligned and right padded. This is most common in BERT applications and mirrors Amrani and Goldberg’s use of BERT. In the second encoding sentences were centered on the target word and padded left and right. The last encoding was also centered on the target word, but the text window was allowed to extend beyond sentence boundaries so that the full sequence would contain text (unless a target word occurred near the very beginning or end of a corpus document).

The BERT tokenizer was used to transform words into word embeddings. BERT word embeddings are vector representations of words that are pre-trained along with the BERT model. These representations were run through the BERT encoder to produce self-attention vectors. The self-attention vectors describe the relationship of words within the text sequence to other words.

This project reasoned that these self-attention vectors could be used as representations for a target word in context and that similar linguistic contexts would share similar self-attention vectors. These vectors were used as input to Growing Neural Gas for clustering.

It was suspected from the beginning that the left aligned, right padded sentences would not be suitable for clustering due to differences in sentence length and position of the target word. When this was confirmed, the centered contexts were produced.

* 1. Machine Learning

Several machine learning frameworks were employed in this project. The first was Pytorch. Pytorch is a popular machine learning library developed by Facebook’s AI Research Lab. Pytorch implements Tensors (multidimensional arrays) that support automatic differentiation. This allows gradients to be computed automatically during backward passes in neural networks. Pytorch also exposes several varieties of premade neural network layers and prebuilt NN architectures.

This project used the Huggingface implementation of the BERT transformer released as part of the Pytorch library. Huggingface has released pre-trained implementations of several popular architectures including BERT and GPT.

Finally, this project also employed Microsoft’s Azure Machine Learning environment. Azure is Microsoft’s cloud computing platform (discussed next). AzureML is a dedicated machine learning environment within Azure. AzureML provides several machine learning tools and models, however none of these were used in this project. Primarily this project used the AzureML framework to execute script runs in the cloud as well as to track runs and their results. AzureML is also capable of cataloging models and datasets and exposing them for use by downstream applications.

* 1. Cloud Computing

This project used Microsoft’s Azure cloud computing platform. Azure was used for several reasons, first among them was scalability. It was originally anticipated that this project would require intensive computing resources. Cloud platforms appeared to be a cost-effective solution to this. Code was developed and tested on local machines and deployed to the cloud for full experimental runs.

Additionally, developing for the Azure cloud platform necessitated careful management of the development environment. A python virtual environment curated under AzureML was adopted and extended to support this project. This environment and all code was deployed to Azure via a Docker container. This method of development ensures that code is portable and that development and execution environments can be recreated on other machines. All code can be run either locally or in the cloud. The same python environment is used in both environments. This portability gives future revisions of this project near unlimited scalability in terms of storage and computing power. Cost will be the only constraint. This project cost $34.55 to produce, test and run on the Azure platform.

* 1. Graph Databases

This project did not directly employ graph databases to support experimentation. However, it was a development goal to persist GNG results to a database.

Azure’s CosmosDB was used to store words, sense inventories and occurrence data. Intermittent GNG results were not written to the database in this version of this project.

CosmosDB is an umbrella term for several non-sql database schemas available on the Azure platform. This project used Azure’s implementation of Apache’s Gremlin / TinkerPop graph database specification.

A graph database platform was chosen to support future development and experimentation. Several WSD approaches including Bevilacqua and Navigli use graph computing approaches to inform sense selection. Additionally, several adaptations of WordNet employ graph databases.

* 1. SEMEVAL 2015

The data used for testing is taken from SEMEVAL 2015 Task 13 Multilingual All-Words Sense Disambiguation and Entity Linking. SEMEVAL is an annual conference on semantic evaluation. It originated as SENSEVAL and was originally focused entirely on WSD; however its’ scope has grown.

The SEMEVAL 2015 Task 13 dataset provides 4 documents, 3 from the biomedical domain and one from social commentary. The SEMEVAL dataset was chosen because it provides a standardized benchmark including pre-made scoring models for evaluating success.

1. Results and Analysis

The current version of this experiment did not return productive results. It became apparent early in testing that clusters were not aligning with hand tagged senses. This was apparent in the number of clusters produced (almost all words collapsed to a single cluster regardless of how many senses were present in the data) and in the Euclidian distance between nodes and the solution sense.

Initially this was thought to result from left aligned, right padded, input. Resultingly, centered contexts were produced and tested, however these inputs also produced unusable clusters. As a result, the SEMEVAL 2015 scoring module was not run as it was evident that the F1 scores would be low.

When positive results were not obtained a single target word was selected for in-depth inspection. The word ‘*use*’ was selected due to its high degree of polysemy. In the corpora keys, use has four different senses associated with it. ‘*use*’ occurs 28 times in the data as a target word. Clustering was run specifically on this subset of the data. The results of clustering mirrored the earlier runs. One cluster with 28 nodes were produced. Success would have been development of 4 clusters.

Clustering was re-run with different hyperparameters. The primary adjustment being number of passes through the GNG network. 10,000 and 100,000 passes were conducted. Both produced the same results. It appears GNG converges on the final network within less than 1000 passes and remains there.

It appears there were several failures in the methodology of this experiment. The most critical was choice of context embedding. The results of clustering indicate that the self-attention vectors produced by the BERT encoder are not spatially similar in n-space. This is a crucial shortcoming as it pre-empts clustering and prohibits sense identification.

It was believed at the outset that the contexts of words with the same sense would be similar enough to support clustering in n-space. It’s evident that wasn’t the case, but further investigation is required to fully determine why.

Although failure of the context embedding prevented collection of meaningful data from the rest of the experiment, several other methodological shortcomings were identified during final analysis.

Again, working on the assumption that words with the same sense would plot in similar locations in n-space, the encodings used for answer key senses were based on the first occurrence of senses in the document. Depending on the results of further investigation of the embedding scheme, this may need to be revised. It is possible an average of all occurrences of a sense might be more appropriate.

With respect to GNG, reflection suggests that GNG clusters and nodes may not be useful in recommending hyponym / troponym relationships due to the movement of network nodes through the input-space. Likely an agglomerative clustering approach would be better suited to this and or a fully graph or knowledge-based approach.

It is still believed that GNG will be successful in mapping sense topology and in detecting changes in word usage once a proper embedding scheme is determined.

On the software side, development of the experiment is considered a success. All development goals were realized. Some code needs to be refactored to support long term reuse and maintenance but over all the system is reusable, extensible and scalable owing to the combination of technologies employed, particularly cloud computing.

1. Conclusion
   1. Summary

This project was a formal first step in exploring computational linguistics. Although the experimental results were not satisfying, the research environment developed will enable rapid reengagement of the topic.

The original aim of the project was to examine the feasibility of scalable granularity in word sense disambiguation. This topic was chosen for its application in downstream semantic processing, specifically disambiguating reference in domain specific text. It is the author’s goal in the future to research document summarization, question answering, and identification of logical contradiction in text.

At the completion of this phase of exploration it remains unclear if scalable granularity in WSI/WSD is achievable as originally envisioned. Going into this project, it was known that other strategies conduct sense clustering first and then use a knowledge-base like WordNet or BabelNet to map to a sense with the proper specificity. However, it was desirable to link a context mapping directly to a sense and conduct both sense clustering and sense identification in one step.

* 1. Next Steps

The next step in this exploration will be to refactor some of the code. In efforts to meet deadlines, not all code was written in accordance with the best practices of Domain Driven Design and Test Driven Development. Cleaning up the code will facilitate long term reuse and maintenance as well as ease future collaboration.

After that, the embedding scheme must be revisited. Further investigation is needed to reassess the applicability of the BERT encoder embeddings as context encodings. If as currently expected they are not, a thorough examination of existing encoding schemes is required to discover or develop the context encoding desired.

Once a suitable context encoding is arrived at, the GNG experiment will be revisited. Agglomerative clustering and iterative application of K-Means clustering will also be considered.

1. References

# & Works Cited

Adaloglou, N. (2021, Nov). *How Transformers work in Deep Learning and NLP*. Retrieved from theaisummer.com: https://theaisummer.com/transformer/

Alammar, J. (2021, Nov). *The Illustrated BERT, ELMo and co.* Retrieved from GitHub: http://jalammar.github.io/illustrated-bert/

Alammar, J. (2021, Nov). *The Illustrated Transformer*. Retrieved from GitHub: http://jalammar.github.io/illustrated-transformer/

Alec Radford, K. N. (2018). *Improving Language Understanding by Generative Pre-Training.*

Aleksandr Slapoguzov, K. M. (2019). *Word Sense Induction for Russian Texts Using BERT.* St Petersburg: ITMO University.

Anaya, M. (2018). *Clean Code in Python.* Birgmingham, UK: Packt.

Andrea Moro, R. N. (2014). *SEMEVAL 2015 Task 13: Multilingual All-Words Sense Disambiguation and Entity Linking.* Roma: Dipartimento di Informatica, Sapienza Universita di Roma.

Apache.org. (2021, Nov). *TinkerPop*. Retrieved from TinkerPop: http://tinkerpop.apache.org/

Arumugan, R., & Shanmugamani, R. (2018). *Hands On Natural Language Processing with Python.* Birmingham, UK: Packt.

Asaf Amrani, Y. G. (2019). *Toward Better Substitution Based Word Sense Induction.* Bar Ilan University, Computer Science Department. Israel: Allen Institute for Artificial Intelligence.

Ashish Vaswani, N. S. (2017). Attention Is All You Need. *31st Conference on Neural Information Processing Systems.* Long Beach.

Bader, D. (2016-2018). *Python Tricks The Book.* dbader.org.

Bengio, Y., Ducharme, R., & Vincent, P. (1999). *A Neural Probabilistic Language Model.*

Diou, C., Katsikatos, G., & Delopolous, A. (2006). Constructing Fuzzy Relations fromWordNet forWord Sense Disambiguation. *2006 First International Workshop on Semantic Media Adaptation and Personalization (SMAP'06).*

Doshi, K. (2021, Nov). *Transformers Explained Visually*. Retrieved from TowardDataScience.com: https://towardsdatascience.com/transformers-explained-visually-part-1-overview-of-functionality-95a6dd460452

Firth, J. R. (1957). *A Synopsis of Linguistic Theory.* Blackwell: Oxford.

Ganegedara, T. (2018). *Natral Language Processing with Tensor Flow.* Birmingham, UK: Packt.

Gudikandula, P. (2021). *Machine Learning for Transformers Explained with Language Translation.* Deep Lobe.

Guille, A. (2021, Nov). *Growing Neural Gas*. Retrieved from GitHub: https://github.com/AdrienGuille/GrowingNeuralGas/blob/master/gng.py

Gunning, D., & Ghosh, S. (2019). *Natural Language Processing Fundamentals.* Birmingham, UK: Packt.

Hugginface.co. (2021, Nov). *Transformers*. Retrieved from Huggingface: https://huggingface.co/transformers/index.html

Jacob Devlin, M. W. (2018). *BERT: Pre-training of Deep Bidrectional Transformers for Language Understanding.*

Learning, M. A. (2021, Nov). *Azure Machine Learning Notebooks*. Retrieved from GitHub: https://github.com/Azure/MachineLearningNotebooks/

Lynn-Evans, S. (2021, Nov). *How to code the Transformer in Pytorch*. Retrieved from Floydhub.com: https://blog.floydhub.com/the-transformer-in-pytorch/

Michele Bevilacqua, R. N. (2020). Breaking Through the 80% Glass Ceiling. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 2854-2864). Association for Computational Linguistics.

Miller, G. A. (1995). WordNet: A Lexical Database for English. *Communications of the ACM*, 39-41.

Navigli, R. (2009). Word Sense Disambiguation: A Survey. *ACM Computing Surveys (CSUR)*.

Popov, A. (2017). Word Sense Disambiguation with Recurrent Neural Networks. *Proceedings of the Student Research Workshop associated with RANLP 2017*, (pp. 25-34). Varna, Bulgaria.

Princeton University. (2010). *"About WordNet".* Retrieved from Princeton University: https://wordnet.princeton.edu/

python.org. (n.d.). Retrieved from Python.org: https://www.python.org/

Python.org. (2021, Nov 19). *Python*. Retrieved from Python: https://www.python.org/

Pytorch.org. (2021, Nov). *Pytorch*. Retrieved from Pytorch: https://pytorch.org/

Raganato, A., Camacho-Collados, J., & Navigli, R. (2017). Word Sense Disambiguation: A Unified Evaluation Framework and Empirical Comparison. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1* (pp. 99-110). Valencia, Spain: Association for Computational Linguistics.

Raschka, S., & Mirjalili, V. (2019). *Python Machine Learning.* Birmingham, UK: Packt.

Reddy Bokka, K., Hora, S., Jain, T., & Wambugu, M. (2019). *Deep Learning for Natural Language Processing.* Birmingham, UK: Packt.

Řehůřek, R. (2019). *GENSIM*. Retrieved from GENSIM: https://gensim.readthedocs.io/en/latest/

Ren, K., & Ren, Y. F. (2015). Kernel Fuzzy C-Means Clustering for Word Sense Disambiguation in BioMedical Texsts. *Journal of Digital Information Management*.

Rossum, G. v. (2021, Nov 19). *A Brief Timeline of Python*. Retrieved from The History of Python: https://python-history.blogspot.com/2009/01/brief-timeline-of-python.html

Scmitt, N., Jiang, X., & Grabe, W. (2011). The Percentage of Words Known in a Text and Reading Comprehension. *The Modern Language Journal*, 26-43.

Scooterlay. (2021, Nov 19). *18 Different Types of Skateboards*. Retrieved from Scooterlay: https://scooterlay.com/types-of-skateboards/#:~:text=18%20Different%20Types%20of%20Skateboards%20to%20Choose%20the,Skateboards-%20Standard%20%20...%20%20%20%2439.99%20

SemEval. (2021, Nov). *SemEval 2015*. Retrieved from SemEval: https://alt.qcri.org/semeval2015/

Sharma, S. (2017). *Activation Functions in Neural Networks.* Retrieved from Toward Data Science: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Srinivasa-Desikan, B. (2018). *Natural Language Processing and Computational Linguistics.* Birmingham, UK: Packt.

Surfingwiki. (2021, Nov 19). *Types of Surfboards*. Retrieved from Surfingwiki: https://www.surfingwiki.com/wiki/Types\_of\_surfboards

Team, P. P. (2019, Dec 17). *The Many Different Types of Board of Directors*. Retrieved from Medium: https://medium.com/process-pa/the-many-different-types-of-board-of-directors-fdf6f5643dae#:~:text=%20The%20Many%20Different%20Types%20of%20Board%20of,advisory%20role.%20They%20are%20similar%20to...%20More%20

Tomas Mikolov, I. S. (2013). *Distributed Representations of Words and Phrases and Their Compositionality.* Mountain View: Google.

Versloot, C. (2021, Nov). *Introductino to Transformers in Machine Learning*. Retrieved from Machinecurve.com: https://www.machinecurve.com/index.php/2020/12/28/introduction-to-transformers-in-machine-learning/

Wikipedia. (2019). *Brown Corpus.* Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Brown\_Corpus

www.nltk.org. (2015). *The Natural Language Toolkit*. Retrieved from NLTK.org: www.nltk.org